Synthesis of Concurrent Implementations from Sequential Ones

Supervisor:
Ahmed Bouajjani, Professor, Univ. Paris Diderot and Institut Universitaire de France.
`http://www.liafa.univ-paris-diderot.fr/~abou/`

Laboratory:
IRIF (Institut de Recherche en Informatique Fondamentale) – formerly LIAFA + PPS

Contact information:
Address: Univ. Paris Diderot, Bat. Sophie Germain, 8 place Aurélie Nemours, 75013 Paris.
Email: `abou@liafa.univ-paris-diderot.fr` *Phone:* `+33 (0) 1 5727 9264`

1 Description of the project

1.1 General context

Abstract data types (ADT) such as sets, stacks, queues, priority queues, etc are implemented using lower level representations such as (doubly-)linked lists, trees, etc. Abstract operations on theses ADT's such *add* an element, *remove* and element, check if the structure is *Empty?*, etc. are implemented by methods that manipulate the low level representations. These implementations are well known when operations are assumed to be executed in a sequential way, i.e., one after the other.

However, in the context of concurrency when several threads can share data structures, implementations of operations on these structures are more difficult to design and to get right. In particular, careful management of ownership of different objects in the data structures should be done in order to avoid undesirable interferences. Several locking policies can be adopted in order to achieve that, allowing more or less parallelism between the threads in accessing the shared data structures. For instance, a coarse-grain locking policy consists in allowing only one thread to access the whole structure at a time. Other locking policies can be more permissive and allow several threads to access the data structures, provided they do not have a visible interference. Then, given a sequential implementation of an abstract data type there are two important issues to consider:

- Verification: for given a locking policy, check that the resulting concurrent implementation is observationally equivalent to the original sequential implementation.
- Synthesis: construct the most permissive locking policy that ensures that the concurrent implementation is observationally equivalent to the sequential one.

1.2 Goal

The goal of this project is (1) to formalize the two questions above, (2) to investigate the decidability and complexity of these problems, and (3) to provide efficient algorithmic solutions for these problems.

1.3 Background

A solid background in formal verification, concurrency theory, automata, and algorithms, will